

机器学习实验报告

院(系):智能工程学院 学号：21312525 姓名：陈佳俊

日期：2022.10.5 实验名称：**线性回归与逻辑回归**

**线性模型**

**基本形式：**

给定由单或多个属性描述的变量**x**，线性模型尝试学得一个通过关于**x**的线性组合来预测的函数，用向量形式可写成：

**线性回归**

**一、实验目的**

1.初步学会使用线性模型，用它处理回归任务与进行预测

2.在应用中尝试学会使用pandas、numpy等强大的数据库

3.学习多种习得单变量线性模型与多变量线性模型的方法，如求导法、梯度下降法、矩阵求解法

4.学会画图，输出可视化结果

5.学会计算均方误差、写出计算损失函数的代码等等

**二、实验思路**

1.准备好训练集和测试集数据，将他们编辑成可用的矩阵变量

2.单变量模型分别使用求导法、梯度下降法、矩阵求解法，多变量模型分别使用梯度下降法、矩阵求解法，求解关于x的线性组合**w**和**b**，结果可以使得损失函数L达到最小

3.使用求得的**w**和**b**与测试数据集中的x组合，预测结果f(x)，并于真实值比较，计算均方误差

**三、实现过程（含部分代码展示）**

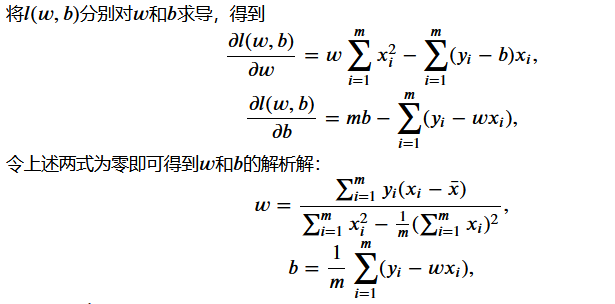
1.读取数据，x和y分别存训练集的属性值和标记值



2.一元线性回归模型

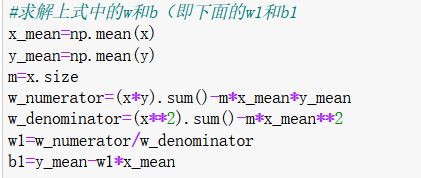
（1）方法①：求导法

原理：



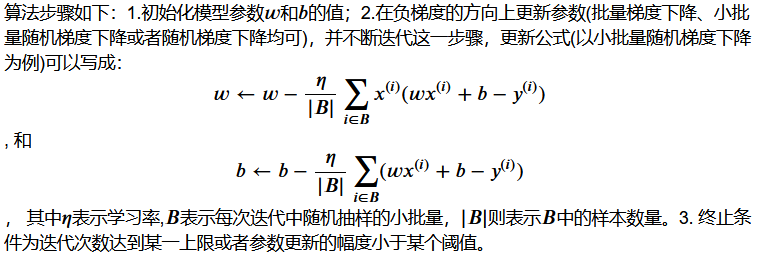
求得的解析解即为使损失函数l达到最小值的解。

实现过程：

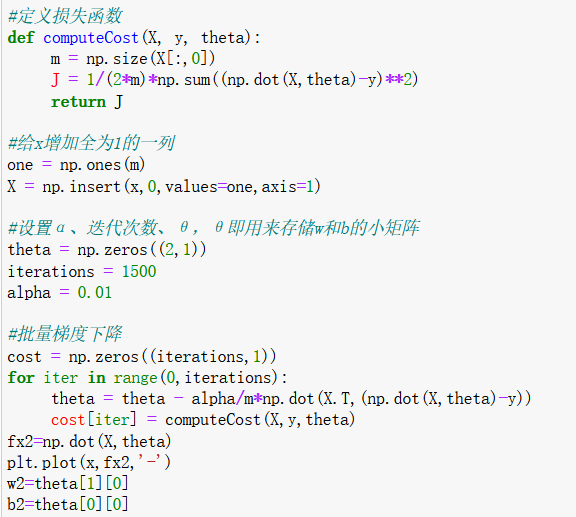


方法②：梯度下降法

原理：



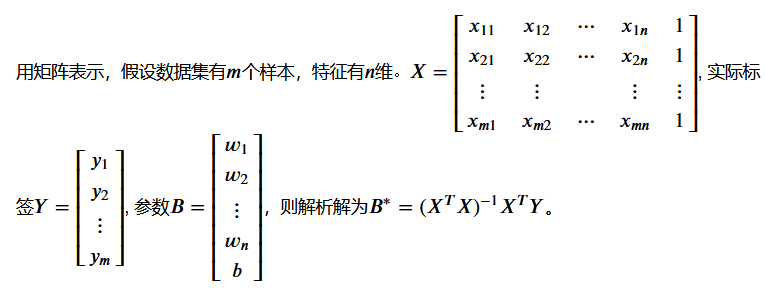
实现过程（以批量梯度下降为例）：



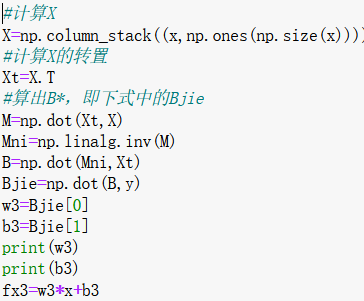
批量梯度下降采用的终止条件是当迭代次数达到一定值时结束迭代。损失函数在这里定义为均方误差，误差与迭代次数的曲线可用于体现结果的可靠性。

方法③：矩阵求解法

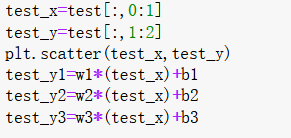
步骤：



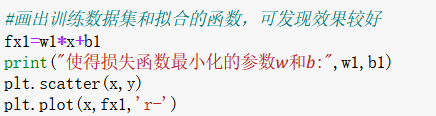
实现过程：

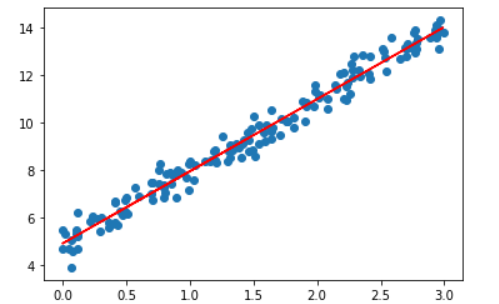


（2）预测数据

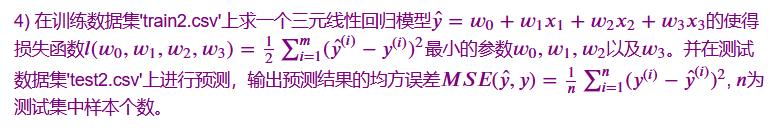


（3）画图（这里以方法①为例）



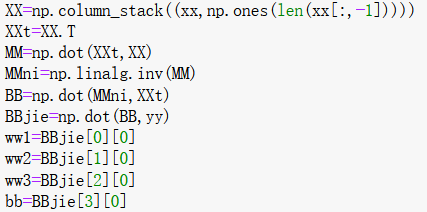


3.三元线性回归模型



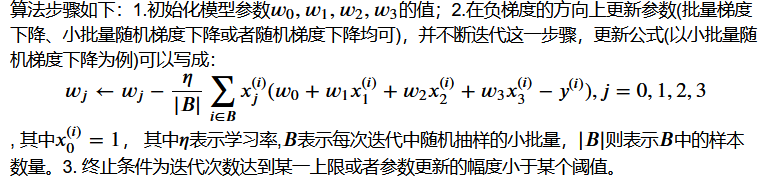
（1）读取数据（与2.中步骤相同）

（2）方法①：矩阵求解法（同2.中方法③）

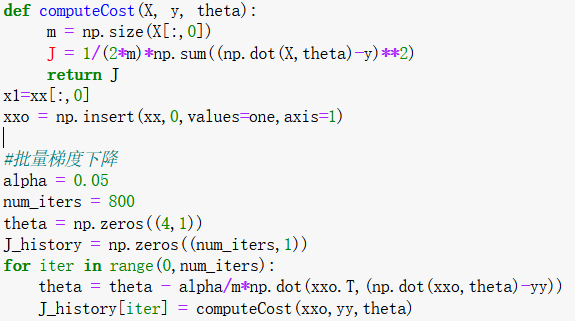


不同之处在于，求得的**w**为三个解，分别对应三个特征。

方法②：梯度下降法（类似2.中方法②）

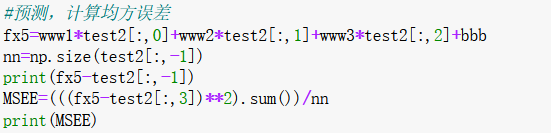


这里我同样使用了批量梯度下降：



其中J\_history可用于画出梯度下降的损失函数随迭代次数下降的曲线，可以直观地验证解的合理性。

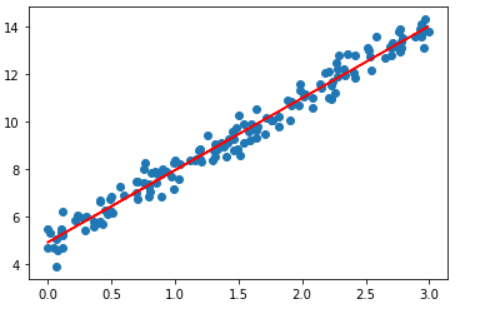
（3）预测并计算均方误差

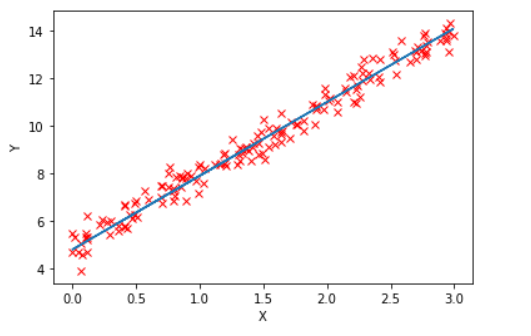


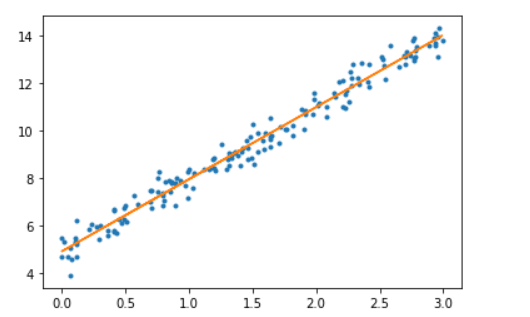
**四、实验成果展示**

1.单变量线性模型：

方法①、②、③分别画出来的直线：



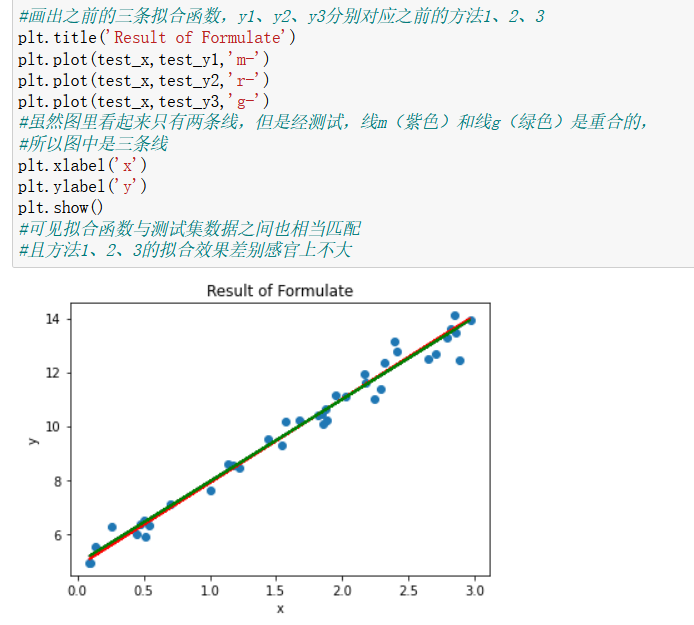




它们的w、b值分别为：

|  |  |  |
| --- | --- | --- |
|  | **w** | **b** |
| 方法① | 3.041 | 4.906 |
| 方法② | 3.105 | 4.788 |
| 方法③ | 3.041 | 4.906 |

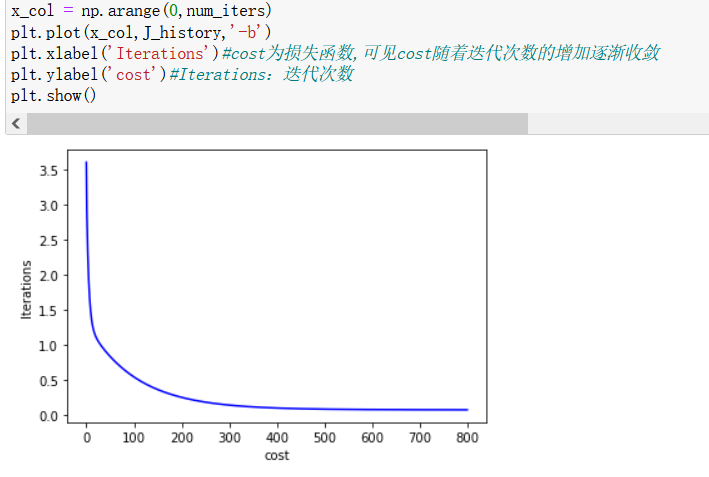
测试数据的散点图以及训练好的模型函数图像：



2.多变量线性模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | W1 | W2 | W3 | b |
| 方法① | 1.007 | 2.003 | 3.010 | 5.943 |
| 方法② | 1.020 | 2.016 | 3.024 | 5.840 |

批量梯度法的迭代曲线：



均方误差MSE（以方法②为例）：0.165

**五、实验细节**

成绩

**六、不足之处**

本实验

**七、实验总结**

输入学生成绩信息序列，获得成绩从高到低、从低到高、按某一门成绩的排列,相同成绩都按先录入排列在前的规则处理。

输出：利用字典容器，完成对于每个学生成绩数据的储存，并将所有学生储存于列表中；在此基础上，按照总分从高到低的学生名单，总分从低到高的学生名单，三门课成绩从高到低的学生名单。

**逻辑回归**

**一、实验目的**

成绩

**二、实验思路**

成绩

**三、实现过程（含部分代码展示）**

成绩

**四、实验成果展示**

成绩

**五、实验细节**

成绩

**六、不足之处**

成绩

**七、实验总结**

**二、实验内容**

1.学生与成绩之间是树形关系，本题学生的成绩共有三个分支，分别是高数、英语、大物，类别不算特别多，因此可以分别用字典去储存成绩数据；

2.将学生数据用列表储存的原因是列表是有序的，好处一是可以在打印不同名单时按照名单的要求对学生列表进行排序，二是可以通过列表中的学生元素作为key值对学生的各科成绩进行访问；

3.题目要求的排序方案是基于比较成绩大小来进行的，因此我使用较为常见的**冒泡排序法**依据学生成绩对学生名单进行排序。

**三、搭建框架与代码实现**

1.分别创建一个用于储存学生姓名的列表和四个用于储存学生成绩（三科与总分）的字典，其中总分可由三科成绩相加得出

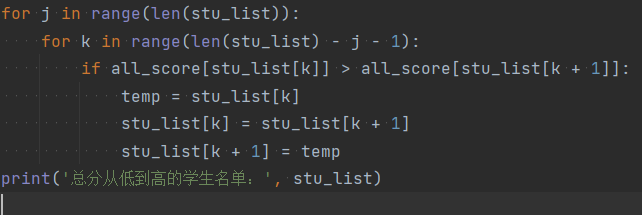


3.之后是按题目要求进行运算，并输出学生名单

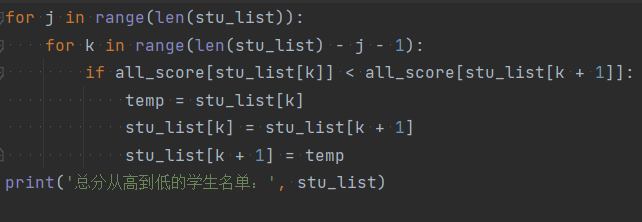
冒泡排序法原理：对数组中的元素进行重复访问，在一次访问中，元素将与位置在它后面一位的元素进行大小比较，如果不符合要求（升序或降序排列），那么两个位置上的内容将被交换，“冒泡”体现在越小的元素会经由交换慢慢“浮”到数列的前端（升序）或后端（降序）。

在题目中的应用：以依据总分升序排序为例，从学生列表中提取学生姓名的数据，从第一个元素开始，然后将姓名作为key值访问总分的字典，提取总分数据，并使用冒泡排序法进行比较。用temp作为交换的临时变量，如果前面的元素比后面的元素大，那么交换它们，排序后的结果将保留在原来的学生姓名列表中。对于其他要求，如降序排列，则将“>”改为“<”，对单科排名，则将总分字典改为单科的字典。

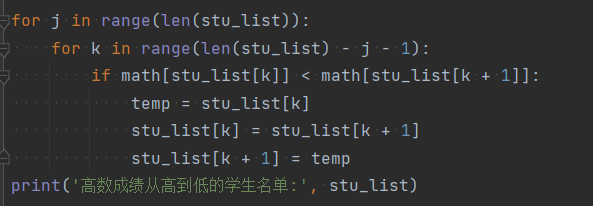
（1）总分从低到高的学生名单



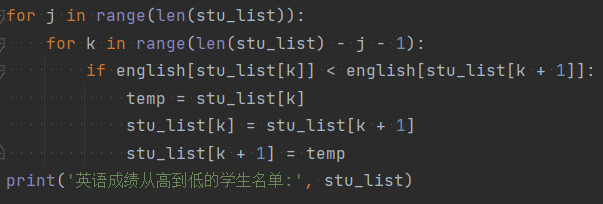
（2）总分从高到低的学生名单



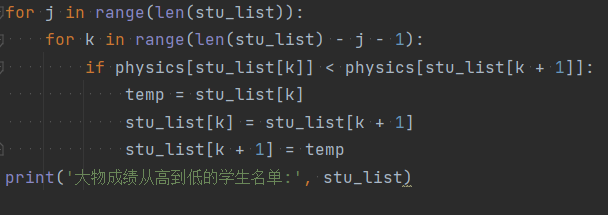
（3）高数成绩从高到低的学生名单



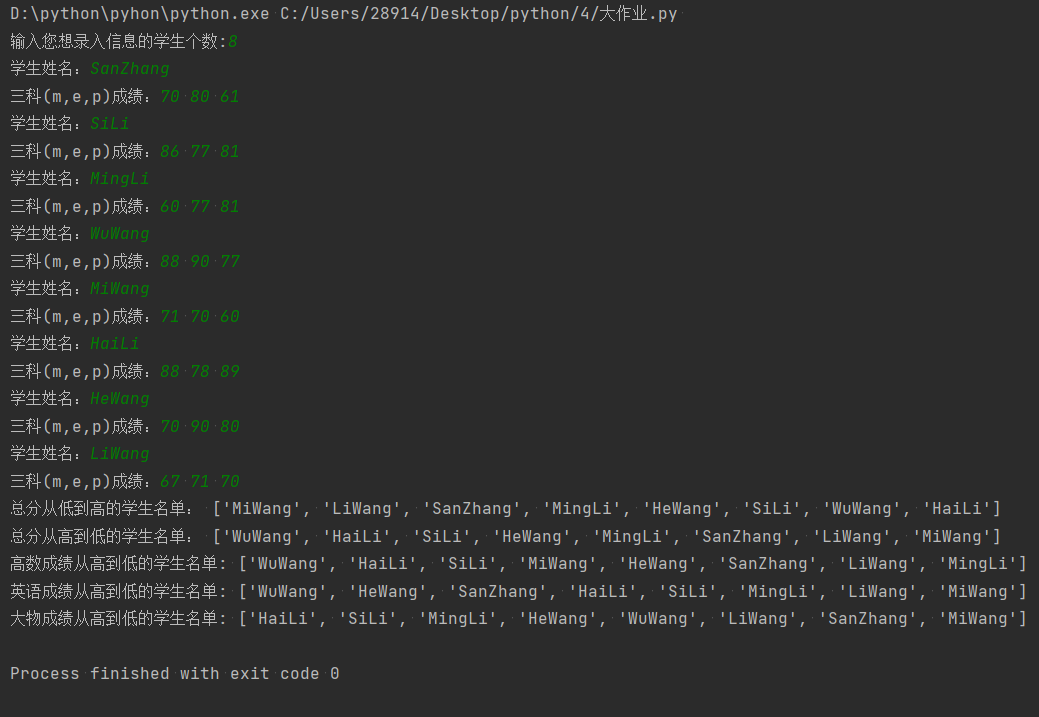
（4）英语成绩从高到低的学生名单



（5）大物成绩从高到低的学生名单



**四、输入与输出的流程**

****

**五、操作细节与待改进之处**

细节：题目要求相同成绩都按先录入排列在前的规则处理，所以我们在冒泡排序中使用的是“>”而不是“≥”，若相同成绩，则不进行交换，可以达到先录入的排在前的目的。此题没有使用快速排列法，也是因为题目的这个要求，快速排列法的不稳定性让先录入排在前的要求难以实现。

待改进之处：我的程序一开始就要求用户指定录入成绩的学生个数，这在实际应用中不一定实用，因此可以把算法改进成能够满足输入某个值（比如“0”）就停止录入，否则继续录入的操作的算法。此外，在打印名单的过程中改变了原录入名单的顺序，省了应用的空间但增加了不稳定的因素。

**六、总结（兼心得体会）**

当得知要写大作业的程序时，觉得很难，但后来开始思索后觉得也挺好上手。在经过一段时间编写后程序也初步成型，之后发现了许多不足之处，比如录入的过程过于繁琐等等，最后将代码进行了修改完善形成了最后的程序。所以个人认为借助python强大的功能完成一个程序并不难，在这次作业中重要的还是对输入、输出操作等细节上的完善，提升用户体验。虽然做了很多功夫完善细节，但最后的效果也有很大的空间提高，比如输出名单可以更美化一些。发现细节是一个永无止境的过程，要看自己对程序的重视程度。

**七、完整代码**

math = dict()

english = dict()

physics = dict()

all\_score = dict()

stu\_list = list()

n = int(input('输入您想录入信息的学生个数:'))

for i in range(n):

stu = input('学生姓名：')

x, y, z = map(int, input('三科(m,e,p)成绩：').split())

math[stu] = x

english[stu] = y

physics[stu] = z

all\_score[stu] = x + y + z

stu\_list.append(stu)

for j in range(len(stu\_list)):

 for k in range(len(stu\_list) - j - 1):

if all\_score[stu\_list[k]] > all\_score[stu\_list[k + 1]]:

temp = stu\_list[k]

stu\_list[k] = stu\_list[k + 1]

stu\_list[k + 1] = temp

print('总分从低到高的学生名单：', stu\_list)

for j in range(len(stu\_list)):

for k in range(len(stu\_list) - j - 1):

if all\_score[stu\_list[k]] < all\_score[stu\_list[k + 1]]:

temp = stu\_list[k]

stu\_list[k] = stu\_list[k + 1]

stu\_list[k + 1] = temp

print('总分从高到低的学生名单：', stu\_list)

for j in range(len(stu\_list)):

for k in range(len(stu\_list) - j - 1):

if math[stu\_list[k]] < math[stu\_list[k + 1]]:

temp = stu\_list[k]

stu\_list[k] = stu\_list[k + 1]

stu\_list[k + 1] = temp

print('高数成绩从高到低的学生名单:', stu\_list)

for j in range(len(stu\_list)):

for k in range(len(stu\_list) - j - 1):

if english[stu\_list[k]] < english[stu\_list[k + 1]]:

temp = stu\_list[k]

stu\_list[k] = stu\_list[k + 1]

stu\_list[k + 1] = temp

print('英语成绩从高到低的学生名单:', stu\_list)

for j in range(len(stu\_list)):

for k in range(len(stu\_list) - j - 1):

if physics[stu\_list[k]] < physics[stu\_list[k + 1]]:

temp = stu\_list[k]

stu\_list[k] = stu\_list[k + 1]

stu\_list[k + 1] = temp

print('大物成绩从高到低的学生名单:', stu\_list)

